A METHOD AND APPARATUS EMPLOYING ONE-WAY TRANSFORMS

CROSS REFERENCE TO RELATED APPLICATION

This application is a continuation-in-part of my earlier U.S. provisional application Serial No. 60/231,526 filed on September 11, 2000.

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to systems and devices that implement and make use of one-way transforms and to apparatuses and methods that realize the one-way property via processes and/or protocols.

Background Description

One-way transforms play an important role in forming the basis for data security. The idea of asymmetric invertible one-way transform was introduced in "New Directions in Cryptography" by W. Diffie and M. Hellman, IEEE Transactions on Information Theory, Vol. IT-22, 1976, pp. 644-654. Since then, many schemes and systems for the realization of asymmetric one-way functions came into being. The RSA cryptosystem is described in U.S. Patent No. 4,405,829 to R. Rivest, A. Shamir and L. Adleman. The cryptosystem of T. ElGamal is depicted in "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms", IEEE Transactions on Information Theory, Vol. 31, 1985, pp. 469-472. The more recently advanced cryptographic systems using elliptic curves started with V. Miller's paper "Use of Elliptic Curves in Cryptography", Advances in Cryptology CRYPTO

'85 Proceedings, Berlin: Springer-Verlag, 1985, pp. 417-426.

OBJECTS AND SUMMARY OF THE INVENTION

It is an object of this invention to provide methods of invertible one-way transforms and to provide means of constructing devices that realize invertible one-way transforms. It is another object of this invention to improve on prior art and to provide better methods of realizing invertible one-way functions.

Encryption and decryption are respectively synonymous with the terms *forward transform* and *backward transform* used in the provisional application literature. Therefore, forward (backward) transform parameters are the parameters making up the encryption (decryption) key.

This invention facilitates *unbalanced* correspondence between encryption keys and decryption keys, where one correspondence defines the association of a single encryption key with many different decryption keys and another correspondence defines the association of a single decryption key with many different encryption keys. The cryptographic keys by this invention are *complete* where, once generated, no additional key parameters nor changes in either key parameters or key parameter values are required for performing encryption or decryption multiple times. Furthermore, the construction of the cryptographic keys of this invention has the potential for high parallelism to offer fast encryption.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Let the functions for generating the encryption and decryption keys be denoted by $\boldsymbol{f}()$ and $\boldsymbol{b}()$ respectively, and a cryptographic transform $\boldsymbol{T}$ using some parameters $\boldsymbol{p}$ by $\boldsymbol{T}_p()$. Then the following can hold for the transforms (i.e. encryption and decryption) of this invention:

for any determinant $\mathbf{D}$ and random input $\mathbf{I}$ and $\mathbf{I}'$, $\mathbf{I} \neq \mathbf{I}'$ if and

only if $f(\mathbf{D}, \mathbf{I}) \neq f(\mathbf{D}, \mathbf{I}')$ and/or $b(\mathbf{D}, \mathbf{I}) \neq b(\mathbf{D}, \mathbf{I}')$, and

for any x that is properly encoded, $x = T_{b(\mathbf{D},\mathbf{I})}(T_{f(\mathbf{D},\mathbf{I})}(x))$

where a determinant is a sequence of properly encoded symbols, the value of which determines, in conjunction with any applicable random input, both the actual cryptographic key parameters and the introduction of random noise.

In one embodiment of this invention, *perfect revelation* is realized through the use of a *secrecy primitive*, an entity associated with two parties who have different knowledge about said entity. In particular, some secret known to one party and securely conveyable to another party is contained in such an entity which itself is not required to be kept secret. By making use of this entity, the two parties can securely establish a second entity that is cryptographically symmetric, i.e. the two parties can share a secret.

In another embodiment, some encryption key parameters are converted to a different representation to facilitate other cryptographic techniques.

In still another embodiment, random noise independent of the value of any other cryptographic key parameter is incorporated.

In yet another embodiment, encryption key parameters are represented in *self-contained* (c.f. next paragraph for definition) components to facilitate independent calculation on these components.

An example is given here for illustration purposes. Let us assume $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is a set of positive integers satisfying: $\mathbf{x}_i > (2^{\mathbf{h}}-1)(\mathbf{x}_1 + \mathbf{x}_2 + \ldots + \mathbf{x}_{i-1})$ for $2 \leq \mathbf{i} \leq \mathbf{n}$, and is transformed to $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ via one or more rounds of invertible strong modular multiplication (i.e. each modulus used is greater than the largest possible subset sum of the set that is being applied the strong modular multiplication). Suppose $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ is

the final transformed version with $\mathbf{t\text{-}k} \geq 0$ noise components, where $\mathbf{t}$ is an arbitrary or random number and $\mathbf{z}_i$ for $1 \leq \mathbf{i} \leq \mathbf{n}$ are vectors of $\mathbf{t}$ dimensions, denoted as $\mathbf{z}_i = (\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \ldots, \mathbf{z}_{i,t})$. Let $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_t$ be $\mathbf{t}$ pairwise co-prime numbers and $\mathbf{J} = \{\mathbf{j}_1, \mathbf{j}_2, \ldots, \mathbf{j}_k\}$ be a set of randomly selected indices such that $\mathbf{z}_{i,j} = \mathbf{y}_i \ \% \ \mathbf{p}_j$ if $\mathbf{j} \in \mathbf{J}$ (where % denotes the modular function), and $\mathbf{z}_{i,j}$ is a random number modulo $\mathbf{p}_j$ otherwise, and that the product of $\mathbf{p}_j$ for $\mathbf{j} \in \mathbf{J}$ is greater than the largest possible subset sum of $\mathbf{Y}$. In essence, $\mathbf{Y}$ is reduced to a residue system with arbitrary or random numbers inserted in arbitrarily or randomly picked dimensions in the vectors. This reduction by $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_t$ can also be multiplicative modular reduction. In such residue system representation, the $\mathbf{z}_{i,j}$'s are *self-contained*, which means that, with regard to pertinent cryptographic operations, computation performed on $\mathbf{y}_i$ can be equivalently carried out with each individual of the $\mathbf{z}_{i,j}$ independently. If we lay out $\mathbf{Z}$, with each of its vector element as a row, we will have a matrix format:

$$
\begin{array}{l}
\mathbf{z}_{1,1}, \mathbf{z}_{1,2}, \ldots, \mathbf{z}_{1,t} \\
\mathbf{z}_{2,1}, \mathbf{z}_{2,2}, \ldots, \mathbf{z}_{2,t} \\
\ldots \\
\mathbf{z}_{n,1}, \mathbf{z}_{n,2}, \ldots, \mathbf{z}_{n,t}
\end{array}
$$

and the random components are the columns of random numbers $\mathbf{z}_{i,j}$ for $1 \leq \mathbf{i} \leq \mathbf{n}$ where $\mathbf{j} \notin \mathbf{J}$. $\mathbf{Z}$ and $\mathbf{p}_j$ for $1 \leq \mathbf{j} \leq \mathbf{t}$ are the encryption key, and are not required to be kept secret.

Let the data stream be assembled into $\mathbf{nh}$-bit blocks with necessary padding of random bits, where each block is further divided into $\mathbf{n}$ sub-blocks $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n$ of $\mathbf{h}$ bits each. A block is encrypted to $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_t$ in the following way:

$$\mathbf{c}_j = (\mathbf{d}_1\mathbf{z}_{1,j} + \mathbf{d}_2\mathbf{z}_{2,j} + \ldots + \mathbf{d}_n\mathbf{z}_{n,j}) \ \% \ \mathbf{p}_j, \text{ for } 1 \leq \mathbf{j} \leq \mathbf{t}$$

The $\mathbf{c}_{j \notin J}$, for the mere purpose of recovering the original data, are simply discarded and ignored. Then the original data block is recovered via the recovery of the individual sub-blocks $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n$. One specific recovery processes is to convert the $\mathbf{c}_{j \in J}$ from the residue

system by the $\mathbf{p_j}$'s using the Chinese Remainder Theorem to a subset sum of **Y** in the normal positional number system, and to then apply the round(s) of inverse strong modular multiplication. Finally, the normal decomposition of a superincreasing subset sum can be used to recover the sub-blocks $\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_n}$.

Another type of one-way transform is carried out through the use of a secrecy primitive. In one embodiment, the method of elimination via a protocol can securely single out from the digitized secrecy primitive bits of interest as shared secret. However, in other embodiments, the shared secret can be established indirectly through the establishment of another shared secret. In the following example, one type of indirect establishment of a shared secret is manifested.

The general idea behind is that two parties, **X** and **Y**, will perform a protocol using a set of encryption keys as a secrecy primitive that may be known to observers. From the execution of the protocol, it is infeasible for an observer to deduce the secret established between **X** and **Y**, even though the observer learns everything of the actual transmissions between the two parties, besides having the knowledge of the encryption keys.

We assume that **Y** has **m** authentic encryption keys $\mathbf{T_1}, \mathbf{T_2}, \ldots, \mathbf{T_m}$ for which **X** has the corresponding decryption keys and can learn about the values of certain bits encrypted. To be specific, we assume that **X** can learn the value of the $\mathbf{t_i}^{th}$ bit encrypted using $\mathbf{T_i}$. **Y** will encrypt random bits using the sets of encryption keys and send the encrypted version to **X**. **X** will instruct **Y** to perform certain actions, such as changing the logical index of the $\mathbf{t_i}^{th}$ bit as in the detailed demonstration that follows. By the end of the protocol, **Y** will be able to learn that **X** intended to convey the bit positions $\mathbf{t_i}$. We assume the random data bit blocks used for $\mathbf{T_1}$ are:

1st data block: 10111010001010110100111011010000

2^{nd} data block: 11101010011110010101110111010100

3^{rd} data block: 10001110101010100101110101010101

4^{th} data block: 01001000111101100110101010011111

5^{th} data block: 01110001011001000101110111011101

6^{th} data block: 10100011011011001010100001110101

We also assume, without loss of generality, that $\mathbf{t}_1 = 11$ and **X** intends to have **Y** logically change the indices $\mathbf{t}_i$, for $1 \le \mathbf{i} \le \mathbf{m}$, to the target logical position 17, where the bit position is zero-oriented, counting from left. At the start, the physical positions and the logical positions are the same:

PP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ILP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Here PP stands for Physical Position, LP stands for Logical Position, ILP stands for Initial Logical Position, and FLP stands for Final Logical Position.

To logically move the bit from the 11^{th} position to the 17^{th} position we need to move right a total of 6 bits. We may randomly express 6 as the sum of **k** integers, i.e. we design it so that after **k** shifts, the 11^{th} bit is logically moved/changed to the 17^{th}. In our example, since we are using 6 data blocks, **k** will be 6, i.e. after 6 shifts we make sure the logical position of the 11^{th} bit is the 17^{th}. We assume that we have $6 = 2 + (-8) + 13 + \phi + 0 + (-1)$, where ϕ is a non-zero integer, functionally non-contributing to the sum (6). It indicates a shift that is not effective with regard to the bit of interest, i.e. the logical shift is done only to bits with value opposite to that of the bit of interest. The following is an example execution of the protocol.

Y encrypts the first data block and sends the encrypted version to **X**.

After decryption, **X** obtains the value of the 11^{th} bit in the data block to be 0. He instructs

Y to logically right shift 2 positions (i.e. equivalently adding 2 to the logical position) all bits corresponding to the bits in the data block having value zero

Recall, the first number in the breakdown of 6 (into 2 + (-8) + 13 + ϕ + 0 + (-1)) is 2 and that is how the right shift of 2 comes about. The physical positions (zero oriented) of the bits in the first data block having value zero are: 1, 5, 7, 8, 9, 11, 13, 16, 18, 19, 23, 26, 28, 29, 30, 31. The logical positions corresponding to those physical positions are incremented by 2 and the resulting logical positions will become:

PP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LP	0	3	2	3	4	7	6	9	10	11	10	13	12	15	14	15	18	17	20	21	20	21	22	25	24	25	28	27	30	31	0	1

Notice that the increment is addition modulo 32, i.e. with the block size as the modulus. In other words, the shift is cyclic in essence. Therefore, the logical positions 30 and 31 become 0 and 1 respectively after the increment.

The physical 11th bit of the second data block (that is encrypted by **Y**) is 1, **X** instructs logical shifting of all one-bits -8 positions (or shifting left 8 positions). The one-bits in the second data block are in physical positions 0, 1, 2, 4, 6, 9, 10, 11, 12, 15, 17, 19, 20, 21, 23, 24, 25, 27 and 29. After logical shifting, the results are:

PP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LP	24	28	26	3	28	7	30	9	10	3	2	5	4	15	14	7	18	9	20	13	12	13	22	17	16	17	28	19	30	23	0	1

Similarly, the results from the third data block are:

PP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LP	24	9	7	16	28	7	30	22	10	16	2	18	4	28	14	20	31	9	31	13	12	13	3	17	7	17	9	19	11	23	13	1

In the fourth round, **X** is to instruct a fake shift (ϕ-shift), one that does not affect the logical index of the bit corresponding to the 11th physical bit. Such an instruction is indicated

by ϕ. After the fourth data block, for which we assume a right shift of 4 (i.e. ϕ=4) for the zero-bits because the 11^{th} bit has value 1, the logical positions become:

PP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LP	28	9	11	20	28	11	2	26	10	16	2	18	8	28	14	24	3	9	31	17	12	17	3	21	7	21	13	19	11	23	13	1

After the fifth data block, none of the logical positions changes as we instructed a zero shift. This is of course an actual no-operation, a waste that can be eliminated in actual practice. It is here, however, to illustrate the functional difference between an actual no-operation and a functional no-operation. Both contributes nothing to (6) the actual positions shifted for the bit of interest (11^{th}), but the ϕ-shift does change some logical indices.

After the last (sixth) data block, the logical positions finally become:

PP	0	1	2	3	4	5	6	7	8	9	10	**11**	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FLP	28	8	11	19	27	10	2	26	9	16	2	**17**	8	28	13	23	3	8	31	16	12	16	2	20	6	21	13	19	10	23	12	1

The logical index value corresponding to the 11^{th} physical position is 17, functionally signifies that the 11^{th} physical position has now 'logically' become the 17^{th} as desired.

The same can be done with the other **m**-1 encryption keys, to move the $\mathbf{t_i}^{th}$ bit logically to the target logical position 17. This can be done either sequentially, one bit block after another, or better still in parallel. When the protocol completes, the logical index 17 must appear in each and every of the FLP rows. The identification process for $\mathbf{t_i}$ is as follows.

For any FLP row, if a certain logical index is missing, that logical index in all other (**m**-1) FLP rows is eliminated. For example in the above example, index 4 is not in the FLP row, then index 4 is eliminated from all other FLP rows. If after this elimination process, there are still more than one distinct logical index not eliminated, which will be very rare if **k** and **m** are chosen appropriately, the protocol can be re-executed or extended with more rounds. In

other words, **k** can be increased with the application of more random bit blocks for each encryption key. When only one distinct logical index is left, the physical index corresponding to the logical index is the one **X** intends to communicate to **Y**. If **Y** again encrypts **m** random bit blocks $\beta_1, \beta_2, \ldots, \beta_m$ using $\mathbf{T_1}, \mathbf{T_2}, \ldots, \mathbf{T_m}$ respectively, **X** and **Y** would be able to share the knowledge of the value of the $\mathbf{t}_i^{th}$ bit in β_i. However, the remaining index in a FLP row could have multiple appearances. For instance, logical index 28 appears in both the 0^{th} and the 13^{th} entries in the FLP in the above example. Should **X** have chosen 28 as the target logical position to shift to, **Y** would still not be able to know if physical index 0 or 13 **X** intended. But this can be easily overcome with other means. Assuming, for example, that the $\mathbf{i}^{th}$ FLP row has more than one physical index corresponding to a logical index, bits of β_i in all those physical bit positions can be set to the same value so that the two parties can always have the same value for the $\mathbf{t_i}^{th}$ bit of β_i.

The above example of one-way transform realized via a protocol gets the one-way property from utilizing a set of encryption keys. Such encryption keys can have more than one distinct decryption keys that decrypt a same ciphertext to different results. One should notice that any entity possessing the authentic encryption keys will be able to execute the protocol with **X**, and an attacker can also compromise the contents of the communication between **X** and **Y**. Therefore, the legitimate communicating parties have to properly identify each other to guarantee that the encryption keys are authentic at party **Y**. Furthermore, they must make sure that their communication is not compromised, by applying data integrity techniques which abound in prior art.

It should be obvious and clear to one skilled in the art that the examples are for illustration purposes only. Parameters and assumptions used in the examples are for the convenience of explanation of this invention. In practice and actual implementation of this invention, proper parameters and parameter values should be chosen to meet the requirements of the applications.